

Provisioning, deployment, and operation of smart grid applications on substation level

Bringing future smart grid functionality to power distribution grids

Mario Faschang¹  · Stephan Cejka² · Mark Stefan¹ · Albin Frischenschlager² · Alfred Einfalt² · Konrad Diwold² · Filip Prösl Andrén¹ · Thomas Strasser¹  · Friederich Kupzog¹

Published online: 19 July 2016

© The Author(s) 2016. This article is published with open access at Springerlink.com

Abstract The transition of classical power distribution grids towards actively operated smart grids locates new functionality into intelligent secondary substations. Increased computational power and newly attained communication infrastructure in thousands of secondary substations allow for the distributed realization of sophisticated functions, which were inconceivable a few years ago. These novel functions (e.g., voltage and reactive power control, distributed generation optimization or decentralized market interaction) can primarily be realized by software components operated on powerful automation devices located on secondary substation level. The effective and safe operation of such software

is crucial and has a broad set of requirements. In this paper, we present a flexible and modular software ecosystem for automation devices of substations, which is able to handle these requirements. This ecosystem contains means for high performance data exchange and unification, automatic application provisioning and configuration functions, dependency management, and others. The application of the ecosystem is demonstrated in the context of a field operation example, which has been developed within an Austrian smart grid research project.

Keywords Future distribution system operation · Intelligent secondary substation · Distributed software applications · Smart grid application hosting

✉ Mario Faschang
mario.faschang@ait.ac.at

Stephan Cejka
stephan.cejka@siemens.com

Mark Stefan
mark.stefan@ait.ac.at

Albin Frischenschlager
albin.frischenschlager@siemens.com

Alfred Einfalt
alfred.einfalt@siemens.com

Konrad Diwold
konrad.diwold@siemens.com

Filip Prösl Andrén
filip.proestl-andren@ait.ac.at

Thomas Strasser
thomas.strasser@ait.ac.at

Friederich Kupzog
friederich.kupzog@ait.ac.at

¹ AIT Austrian Institute of Technology GmbH,
Donau-City-Straße 1, 1220 Vienna, Austria

² Siemens AG Österreich, Siemensstraße 90, 1210 Vienna,
Austria

1 Introduction

As analyzed by Rifkin [16], industrial revolutions have all come along with an evolutionary step in information and communication technology (ICT). This also applies for energy systems and especially for electric energy systems. Given the ongoing Internet of Things (IoT) evolution, each and every power grid component can evolve from a former passively or manually operated device into a communicating and interacting element [6, 10, 23]. IoT-enabled power grid devices have already reached the substation and distribution grid level (e.g., smart meters, smart breakers, electric vehicles, smart storage systems) and allow for an active distribution grid operation with the goal of optimizing the power system to guarantee cost-effective and CO₂-neutral operation. However, smart grid IoT components alone are not sufficient to reach these high level goals. In addition sophisticated functions and services are required, which make use of the ICT-connected power grid components, and intercon-

nect them with external services (e.g., weather forecast, user integration), therefore creating new applications for smart energy systems.

Within the introduction we will present the current state of power distribution grid control and discuss new applications for smart energy systems. Subsequently, we will identify the problem statement regarding an IoT-compliant smart grid software ecosystem. In Sect. 2, user stories of the most relevant stakeholders are presented. Based on the user stories key requirements will be derived in Sect. 2.2. Section 3 shows the architecture of the software ecosystem that allows for convenient deployment, provisioning, and operation of smart grid applications. Finally, we demonstrate the functionality of the designed and implemented ecosystem on an exemplary application of the Austrian research project *SCDA—Smart City Demo Aspern* [1] in Sect. 4. The paper closes with a conclusion and an outlook regarding the further application the presented work in Sect. 5.

1.1 Current state of distribution grid control

Currently, low voltage distribution grids are primarily passively operated and hardly have any active control components. In this section we describe the current state of power distribution grid control and automation system components in typical distribution grids in Austria, followed by the current state of development and the actual situation in international research projects.

1.1.1 Control systems and their software in Austrian distribution grids

The degree of automation of today's power distribution grids is quite different. Usually distribution grids are highly automated to substation level and operated by supervisory control and data acquisition (SCADA) systems [18]. Modern SCADA systems are enhanced by a distribution management system (DMS) which offers supporting tools for network analysis, fault location detection or Volt/VAR Control. Many distribution system operators (DSOs) are now beginning to automate medium voltage cells in secondary substations as well to increase systems' reliability and recovery time in case of failures. Low voltage grids are usually not automated and there are hardly any measurements available. A dedicated operation of low voltage grids is not state of the art today, thus it will be a challenge in near future to operate them with a high penetration of photovoltaic systems, batteries, and electric vehicles. Therefore, it is crucial to evaluate promising concepts for low voltage grid operation tools and processes, and how they interface with medium/high voltage SCADA.

1.1.2 Current state of automation and control systems development

In order to fulfill future needs power distribution grids are currently in a transformation into active grids; medium voltage as well as low voltage grid components (smart transformer stations, inverter-based distributed energy resources, smart breakers, intelligent charging stations, etc.) are equipped with remote control functions and monitoring possibilities [13, 18, 19]. Intelligent electronic devices (IED)—kind of microprocessor-based embedded controllers used in power system equipment—offer such enhanced and necessary functionalities [3]. Comparable approaches have already been applied in the past denoted under the term remote terminal unit (RTU). Those devices are usually connected to SCADA systems and offer similar remote control and monitoring functions to energy utilities. However, the communication and control of RTUs was mainly carried out in a proprietary way by using communication protocols and field bus approaches provided by large vendors. Smart grid applications require open, interoperable, and scalable systems to support solutions of low engineering efforts [3, 19]. Proprietary automation and control concepts may partly support these requirements.

To overcome shortcomings related to interoperability, standardization bodies like the IEC proposed the usage of open communication concepts and corresponding data models for interacting with intelligent components and SCADA systems [3, 18, 19]. The most promising approaches nowadays are the IEC 61850 interoperability definition for IEDs as well as the common information model (CIM) used for power grid operation. Since they are not covering implementation languages, other approaches have to be used, e.g., the industrial automation specification IEC 61131 for programmable logic controllers or the IEC 61499 reference model for distributed, embedded control. Furthermore, interoperability technologies like OPC UA get more attention for the realization of smart grid solutions nowadays [3, 4].

1.1.3 Distribution grid control systems in international research projects

New operation concepts taking future requirements into account are subject to several international research projects. In order to cope with a huge amount of smart grid components, as mentioned above, distributed architectures and control approaches are in the focus of those activities [18, 19, 21, 23]. For example, the European ELECTRA IRP focuses on a highly innovative cell-based operational approach [14]. Corresponding voltage and balancing/frequency control algorithms are being developed which are executed in a distributed manner on cell but also on component level. In the FREEDOM project from the US a new power distri-

bution infrastructure is being developed allowing a Plug & Play integration of intelligent grid components (solid state transformers, intelligent switches) and therefore enabling bidirectional energy flow [12]. Similar to the ELECTRA IRP approach, FREEDM also develops a distributed control solution. Thus, an “Energy Internet” or “Internet of Energy” can be realized. Further interesting approaches focus on microgrid level where often multi-agent control solutions are being implemented [21]. Another remarkable project is IMC-AESOP which developed a cloud-based SCADA using web services which has been also applied to energy systems [5].

Summarizing, distributed control approaches, corresponding architectures, and interconnected IEDs are nowadays more often applied in research and demonstration projects as the traditional, centralized and rigidly coupled SCADA/DMS.

1.2 New applications for smart energy systems

In previous work (cf. [8]) we already proposed a generalized greenfield architecture concentrating on the applications that will be executed on smart secondary substations and other entities, covering a wide set of services in the smart grid, smart building, and smart ICT domains. The greenfield architecture was developed using the smart grid architecture model (SGAM) [3].

Focus of the SCDA project is to enable demand flexibility for energy markets and distribution grids, and to provide plug-and-automate functionality for grid automation and improved distribution grid monitoring [8]. The identified interacting services form an abstract architecture within the SGAM high-level layers, i.e., the Function Layer and the Business Layer (cf. Fig. 1). As a consequence in the greenfield architecture no concrete realization has been proposed.

The concrete realization and implementation (Information Layer, Communication Layer, and Component Layer) are subject of this article. However, since we aim for a universal framework, the concrete top-level applications step behind their supporting functions. We will focus on the software ecosystem of the automation component in the intelligent secondary substation node (iSSN—[9]). Such iSSNs, i.e., secondary substations with monitoring and intelligent devices, shall enable functions of modern energy systems, but are however not yet state of the art for distribution system operators. An architecture overview of such an iSSN including the embedded software ecosystem is given in Sect. 3 (cf. Fig. 2). The operation of an exemplary application with details about the modules’ functionality and their data exchange is presented in Sect. 4 (cf. Fig. 6).

1.3 Problem statement

The scope of the proposed smart grid software ecosystem is to enable a range of novel smart grid ancillary services such

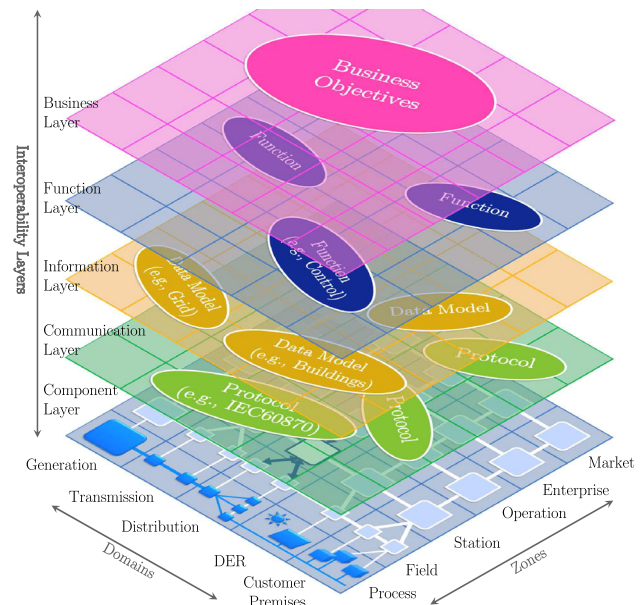


Fig. 1 In the SGAM, the iSSN application frame (cf. [9]) is operated on the component layer’s distribution domain in its (sub)station zone. It serves as a hosting framework for distribution grid-related software application

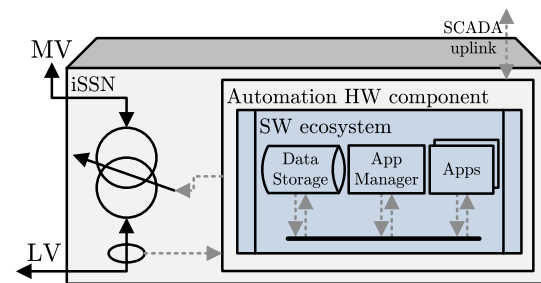


Fig. 2 Architectural overview showing how the software ecosystem is situated on the automation hardware component within the iSSN. It interacts with hardware components in the secondary substation (e.g., tap-changer) as well as with local and remote field sensors and actuators. A SCADA uplink allows an integration in the DSO’s management system

as voltage and reactive power control, distributed generation optimization, decentralized market interaction, electric vehicle charging and storage control algorithms. Such services are based on intelligent and interconnected smart grid entities, e.g., smart meters, smart breakers, electric vehicles, smart storage systems, and smart buildings. In order to achieve these functionalities by smoothly integrating the future smart grid entities, a number of system features are essential.

- *Adaptability of communication* The communication requirements of smart grid applications are manifold [10, 22] and can vary depending on the application domain. A smart grid software ecosystem must therefore provide the means for an easy and seamless integration of new requirements (e.g., new communication stacks).

- *Data storage and processing* All of the above mentioned services depend on information which is acquired from the grid via various devices (e.g., smart meters, sensors, smart buildings). Means of storing, processing, persisting, and accessing such information in a scalable and decentral way is thus a key aspect for the realization of smart grid functionality within a smart grid software ecosystem [23].
- *Plug & Automate functionality* The number of connectable devices and functionalities within low voltage grids is steadily increasing. With this increase in scale manual integration into system operation is not feasible. A smart grid software ecosystem must therefore provide means to minimize the engineering efforts. These include mechanisms for automated application provisioning and hosting as well as methods for the remote and automatic configuration and update of services [19,23].

These system features must be realized by the automation components (primarily in the power distribution domain). Therefore, powerful computational components—which are readily available on the market—have to be equipped by an elaborated and modular software ecosystem that has to be developed.

1.4 Methodology

The methodology for the development of the substation automation component's software ecosystem follows a software engineering approach with respect to the electric energy and distribution grid domain, respectively. The development is based on the services defined in [8] as well as three elementary user stories (from the perspective of a smart grid engineer, a smart grid operator, and an iSSN administrator). Based on this information a set of key requirements is derived and a basic software architecture with key components is proposed. The concrete implementation is developed with respect to the substation's hardware components on which the software ecosystem will be deployed in the field. We evaluate the designed system and its whole functionality by an example of a future smart grid application.

2 User stories and requirements

To support the definition of system requirements, three roles representing areas of responsibilities within a DSO are introduced below and requirements are derived subsequently.

2.1 Smart distribution grid user stories

Relevant roles in the context of Smart Grid Application deployment, provisioning, and operation are the Smart Grid Engineer, Operator, and iSSN Administrator.

2.1.1 Smart grid operator user story

The system operator is responsible for a secure, efficient, and reliable grid operation. The power supply system is a highly critical infrastructure. It has to be ensured that the smart grid operator is not confronted with the increase of complexity in future distribution system operation. He expects aggregated information on the operational state of iSSNs and he should be protected from alarm flooding, e.g., in case of a thunderstorm. If an iSSN reports a problem, it should be easy to decide whether the administrator has to intervene or the engineer directly gets a ticket by the work order management system. The smart grid operator wants to use sophisticated functions (e.g., automated detection algorithms, inter-grid optimization methods, automated topology adaption, graceful degradation measures) to ensure cost effective and reliable operation despite the increase in system complexity. These functions should be available from a well known repository, be auto-configurable to the actual grid, and have a modular and extendable structure.

2.1.2 iSSN administrator user story

The iSSN administrator constitutes a new role (e.g., a grid planner with additional qualification), responsible for installation, update, upgrade, and operation of iSSNs. To fulfill this role he needs the requirements as well as actual information from the grid operator and additionally support by the grid planning group. He expects a user friendly administration environment to keep iSSNs up to date and deploys necessary applications with released parameterization.

2.1.3 Smart grid engineer user story

The smart grid engineer is responsible for maintenance within a defined grid area. If a manual switching operation is planned or in case of a grid-fault the operator uses a work order management system to initiate a task for the maintenance team. He needs support to guarantee a safe switching procedure or to locate and fix a failure efficiently. The iSSN should support the engineer with actual measurements and system status information regarding these tasks. The fast response of iSSN applications, e.g., after grid topology changes, allow a direct and local feedback regarding the outcome of adopted measures.

2.2 Requirements for efficient application deployment, provisioning, and operation

From the above listed user stories, the following requirements are drawn for the development of the software ecosystem of the substation's automation device. As in the course of time

new actors may appear, new requirements may appear and a flexible and extensible system structure is required.

R1—distributed application Expected is a decentral/distributed solution fostering the Plug & Automate paradigm that allows the distributed execution of various software components and their linking by providing reasonable communication mechanisms. A modular software system allows easier extensibility and integration of applications. In critical infrastructures, resilience is a must, keeping influences of one application on others at a minimum.

R2—field component interaction For efficient task fulfillment, quick field component interaction of applications is required and thus, a direct access shall be available. Furthermore, it is required to provide gateways to peripheral components.

R3—automatic deployment Decentralized operation at the secondary substations in the low voltage grids is supervised by a central control center. As such, potentially a high number of substations are required to be equipped with applications providing certain functionalities. Therefore, a mechanism is required, which allows for the automatic deployment of such applications at remote substations without any on-site engineering.

R4—dependency management Some applications may have dependencies on other applications in the form of data or services. This requires a mechanism to automatically resolve dependencies during the installation process of those applications.

R5—versioning Tightly connected with R3 and R4 is the need for an upgrade mechanism to allow the replacement of running applications with a newer version.

R6—automatic configuration Each of the mentioned applications' installations are accompanied with a default configuration. It is however necessary to automatically reconfigure them from remote sites, independent of whether they are already running on the substation.

R7—application monitoring and alarming Local monitoring applications and problem detection algorithms shall allow countermeasures on occurring problems. The state of the substation shall automatically be communicated in real-time to the smart grid operator who can decide to execute further steps such as the installation of additional applications (cf. R3).

R8—high performance data and time series access The majority of conceivable applications require access to historical and current data such as measurement values of smart meters or other sensors. It may be necessary to provide concurrent access to a large amount of data (e.g., one month) for some algorithms. An optimal support for usual data require-

ments in smart grid applications such as large volumes of data, small immutable data records, frequent readouts, and statistical indicators at a local store on substation level has to be provided.

3 Proposed architecture

In order to fulfill the previously identified requirements, the following architecture of the software ecosystem is proposed.

3.1 Architectural overview

Figure 2 presents an architectural overview of the software ecosystem (in blue) and how it is operated on the automation hardware component. The dashed lines show the communication with other iSSN hardware components (e.g., by IEC 61850 or Modbus to the transformer's tap-changer), local measurements at the bus bars, and connections to the PLC-based field communication. Furthermore, a communication uplink to the DSO's SCADA system must be provided.

Concerning software modules, the proposed ecosystem has to provide means of local data storage, an application management system, and a communication middleware to connect these modules with the applications that are intended to be operated within the software ecosystem. The automation hardware component needs to be an industry grade computer with local storage, extended I/O interfaces to support, e.g., IEC 61850, IEC 60870-5-104, Modbus, RS485 and other industry related local and remote communication systems.

3.2 Software modules and processes

The single modules of the substation's software ecosystem are described subsequently. These building blocks operate locally on the industry grade PC and interact by exchanging messages through the Gridlink middleware.

3.2.1 Gridlink

The central component of the use case needs to be a building block that allows communication of several attached applications (cf. R1). Use case requirements were transformed into a specification for a proper communication infrastructure in an iSSN [9]. The implementation of this specification is the Gridlink, a decentralized distributed message bus developed in Java and based on vert.x and Hazelcast. Gridlink was briefly mentioned in previous publications (e.g. [9]) but has not yet been introduced in detail. It is the successor of an earlier platform [7], improving modules' coupling with new functions like a service registry and enabling provisioning features. Various modules, each providing certain features and some of them introduced later in this paper, dynami-

cally form a cluster of known instances during execution [9]. By default, modules running in the same sub net form a Hazelcast cluster, being discovered by using multicast messaging [11]. The modularity of the solution is evident by allowing modules to join or leave at any time without introducing any influence on other modules and their execution. Furthermore, by using a decentralized architecture and not introducing any single point of failure, Gridlink shows to be resilient and scalable [9]. Hence, a fail of any module neither prevents other modules to be executed further nor to communicate with remaining modules. In hand with this paper, the use of Gridlink is now shifted from previous experiences made in laboratory projects [17] to use cases in real world environments.

Modules A module is a Java program that (1) implements a dedicated functionality, (2) takes over one or more roles/topics, (3) is addressable and reachable over the event bus via one (or more) role/topic address(es), (4) provides functions to other modules. In terms of message transmission between modules, every module can serve as a *data source* by issuing messages to other modules by specifying their destination's address and every module can serve as a *data sink* by registering a message handler for these messages. While (1) is fulfilled by all modules, (2)-(4) are not necessarily as modules serving as a *data source* only do not need to take over roles/topics and are thus not addressable. Each module has access to a distributed list of all modules that are currently attached to the Gridlink and active—the *Gridlink Registry*. It includes all roles/topics the module is registered to and a list of requests the according role/topic is able to handle. Modules requiring any communication channel to peripheral components outside the Gridlink system are called *gateway* modules. Currently, implemented use cases require such modules for REST calls, for XMPP protocol handling and for receiving measurement values via an IEC 60870-5-104 translator module (cf. R2).

Direct sending and publish/subscribe mechanism Gridlink supports three types of message exchange; one of these used for implementing a publish/subscribe mechanism, two for direct addressing of modules' roles. Of the latter ones, one is able to register a handler for receiving and handling replies on the issued message, the other one is not. These methods are named *publish*, *sendWithTimeout* and *send* and will later on only be named by these. While being able to address all modules implementing a specified topic by using *publish*, the other two options only address exactly one module, being chosen by round-robin fashion if two or more modules implement the same destination role.

Gridlink proxy The proper execution of sending a message over the Gridlink and receiving this message at the other module is sometimes required to be amended. By using a Gridlink

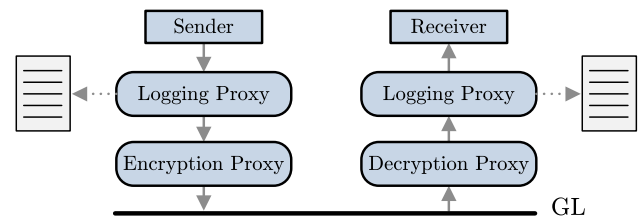


Fig. 3 Example, using one proxy each for logging and message encryption at the sender module, one decryption and logging proxy at the receiver module, respectively. Other examples include a rerouting proxy to change the receiver or a mapping proxy if two modules cannot use the same nomenclature for data points, e.g., for legacy reasons

proxy, the normal execution is interrupted to run user-defined code for interception or modification of messages to be sent or to execute additional tasks when messages are transmitted over the component.

Examples, as shown in Fig. 3, include the logging or encryption of messages. Further examples concerning interception of messages include the filtering of messages, e.g., for a simulation of missing values. More than one proxy can be configured to be run in a serial fashion. Whether a proxy or interceptor is in use, stays transparent to sender and receiver module. It is required to add and remove proxies and interceptors dynamically during the module's execution as described later in Sect. 3.2.2.

Introductory use case example This simple use case consists of three modules: (i) a *data generator module*, periodically producing (random) measurement values, (ii) a *storage module*, for permanent persistence of measurement values, and (iii) a *monitoring module*, that is able to show the most recent or a time span of most recent values.

Listing 1 The storage module registers itself for handling requests on role *storage*. Modules interested in measurement values may not be limited to this single module. We therefore use an event handler for the topic *measurements*, used by all modules eventually interested in such events and to which events will be published to.

```

1 Map<Class, RequestHandler> reqHandlers;
2 Map<Class, EventHandler> evHandlers;
3 ...
4 reqHandlers.put(
5     CreateDataPointRequest.class,
6     new CreateDataPointRequestHandler());
7 reqHandlers.put(
8     GetMostRecentEntryRequest.class,
9     new GetMostRecentEntryRequestHandler());
10 evHandlers.put(
11     MeasurementEvent.class,
12     new MeasurementEventHandler());
13 registerHandler("storage", reqHandlers);
14 registerHandler("measurements", evHandlers);
  
```

Listing 2 The data generator issues a *CreateDataPointRequest* to role *storage* with a new *DataPoint* as payload having name "meterA.u1" and tag "voltage". While in real applica-

tions it would be appropriate to receive a *SuccessReply* on a successful creation of the data point at the storage module or an *ErrorReply* otherwise, the example here should serve as a demonstration for sending messages without expecting any reply.

```
1 Request req = new CreateDataPointRequest(
2   "storage",
3   new DataPoint("meterA.u1", "voltage"));
4 send(req);
```

Listing 3 The data generator periodically issues *MeasurementEvents* to all modules registered to the topic *measurements*, having the measurement as payload.

```
1 double value = ...;
2 Entry measurement = new Entry(
3   "meterA.u1", value, time);
4 publish(new MeasurementEvent(
5   "measurements", measurement));
```

Listing 4 The monitoring module intending to show the most recent value of the data point periodically issues *GetMostRecentEntryRequests* to the storage, having the required data point as payload. Naturally, this requires a reply containing the requested value, hence *sendWithTimeout* is used specifying a code block to be asynchronously executed after the intended reply was received. By Java 8 lambda syntax, it is specified that method *doSomething* is executed after a *GetMostRecentEntryReply* was received. If the specified timeout of 5 seconds expires while no reply has been received, the code block is unregistered and – as no specific error handler was specified – the default error handler is called to react.

```
1 Request req = new GetMostRecentEntryRequest(
2   "storage", new DataPoint("meterA.u1"));
3 ReplyHandler replyHandler =
4   new ReplyHandler(
5     GetMostRecentEntryReply.class,
6     reply -> doSomething(reply));
7 sendWithTimeout(req, 5000, replyHandler);
```

3.2.2 Application provisioning

Application provisioning stands for dealing with remote install, update, upgrade and configure operations to enhance or modify the functionality of a iSSN (cf. R3–R6). Within Gridlink, a designated core module, the *AppManager*, is responsible for receiving provisioning requests (e.g., over REST or XMPP, usually triggered by a user via a web client) and handling received requests. Gridlink modules that are started, configured, updated and stopped by the *AppManager* are called *managed* modules. As the *AppManager* itself is a normal Gridlink (gateway) module, it is able to communicate with any other module. Each module implicitly implements a shutdown role, on which the *AppManager* is able to issue remote shutdown requests. Provisioning tasks are transparent to managed modules. The requirement of not having any

single point of failure still is valid though using the *AppManager*, as its fail crashes the remote provisioning features only. No interference with the operation of other modules arises.

Module installation The *AppManager* receives a request to install a module via its REST or XMPP connection (cf. R3). It downloads a ZIP archive file containing the software to install from an *App Store* and unpacks it to a destination directory accompanied by its default configuration file.

Module start The installed software artifact is started by the *AppManager* by creating a new process (cf. R3). A module usually should get implicitly started after the installation's success. However, there are use cases where this is not the desired way to go. An example is the bulk installation of many modules that have dependencies on each other (cf. R4). Note, that by dependency we do not mean that a module's execution would depend on the other module—this would introduce restrictions on the resilience requirement (cf. R1). However modules, e.g., that require to persist any data will state that they require a storage module to be available. In that case it is desired that a module is not started in hand with its installation, but explicitly started at a later time by using the start command.

Module stop A running module can be stopped by the *AppManager* on receiving a request from the periphery by issuing a Gridlink *ShutdownRequest* to the module's implicit *shutdown* role. The module gets informed that its shutdown was initiated and can react accordingly or can also decide that it is currently not safe to shut down. Therefore the requestee issues an according Gridlink reply including its decision. A negative decision is communicated to the operator user, who can decide to insist on stopping the module immediately. In that case, the *AppManager* who maintains a list of its started modules can force the module to stop if necessary, e.g., by killing the process using its PID.

Module deinstallation Concluding a module's life cycle, a deinstallation command initiates the removal of a stopped module's data from the directory by the *AppManager*.

Module update The module update refers to the replacement of existing running software by another version and is therefore a combination of the described tasks *stop*, *deinstallation*, *installation* and *start* (cf. R5). Therefore, the *AppManager* moves the new artifact to its destination directory. It issues a *ShutdownRequest* to the running module, which can persist its current state to a file. The *AppManager* moves this state file to the directory of the new module and starts it. On start-up the module gets the old state and can continue its work accordingly.

Module configuration The configuration of modules can be altered during their execution (cf. R6). To that purpose, the *AppManager* replaces the configuration file in the module's

directory with the received new version. The module in execution gets informed of a configuration change and may react accordingly; by design it is nevertheless not required to restart the module.

We introduced the *Gridlink Proxy* functionality earlier in this paper and claimed that it should be possible to add and remove proxies dynamically during the module's execution. The implementation of these dynamics are achieved by using described module configuration feature. During the run-time of the module, the iSSN administrator may externally define which proxies should be in service. The replacement of proxies happens without requiring a restart, and transparently to the executing module.

Module information Information of the running modules, such as the configuration that are currently used and their state can be requested to be transmitted over the gateway functionality of the *AppManager* (cf. R7). Described functionality refers to a *pull* approach. It may be suitable to use a *push* approach instead to inform peripheral components on a configuration change.

Planned next steps While it is easy to communicate to modules running on different machines in the same sub net by using Gridlink communication (cf. R1), the *AppManager*'s tasks get more complicated. Coping with file operations, starting and killing modules requires to involve a SSH client. In our use cases it is currently not required that modules run on a different machine. However, we are working on a multi-host concept. In further steps, the *AppManager* may be able to decide where to install a module, e.g., based on the machines' load.

3.2.3 Storacle

Storacle is a Java-based embedded data store for time-series of measurement data and meta data of such generating data points. Key performance indicators vital for Smart Grid infrastructures like high volumes of data of which records are small and immutable, frequent readouts and statistical indicators are optimally supported [2] (cf. R8). In [2] we compared Storacle with state-of-the-art off-the-shelf NoSQL and SQL data bases by using relevant benchmarks and taking into account the limitation of storage size and processing resources that may be present at machines in a substation. A format evaluation in [2] suggested the use of the Protocol Buffer format [15] as basis as it leads to required data size and retrieval time superior to other potential data bases in this use case. In [2] we further listed Cube, RRD4J, Cassandra, InfluxDB, neo4j and OpenTSDB and described why it is not recommended to use these already existing time series database systems for this use case.

The architecture of Storacle, shown in Fig. 4, is based on a three-tier approach. In the first step data to be added

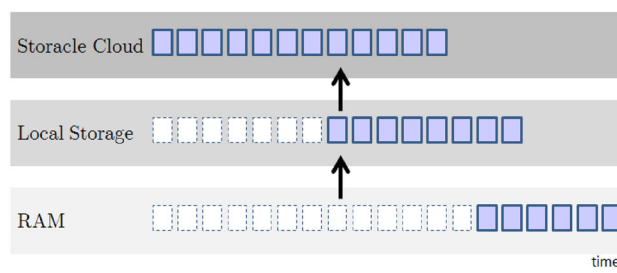


Fig. 4 Storacle Architecture [2]. Currently available records in each layer are shown as *filled boxes*. White *dashed boxes* represent records which have been transferred to a higher layer and are therefore no longer required. Data present at a lower layer, but not yet locally persisted nor copied to the cloud are not shown at higher layers. The *rightmost box* at RAM layer is the most recent entry. *Arrows* represent the data flow to higher layers

is saved in the RAM. Subsequently, data is persisted in a periodic manner to the hard disk. However it will remain in the RAM for some time in order to provide fast responses when queried, reducing costly read-ins from the disk. Optionally, data is periodically copied to a remote location, being a “cloud” or a remote file server which is assumed to have no data size limitations and which may provide replication. After exceeding a defined threshold time, where historical data is no longer of relevance for any local applications and thus these applications do not require access to the data anymore, data can be removed from the local disk once it has successfully been transferred to the third layer for permanent storage. Transfer and removal of old data from the local disk conserves disk space. Depending on the number of data points that need to be persisted, the frequency of their production of data and the available file size, Storacle is capable of persisting long periods of time without using such a third layer. Benchmarks showed that a time series record requires about 18 bytes of storage; one data point producing data with a frequency of 1 Hz therefore requires about 541 MB for one year of data (~31.5 million entries). Reading data back from the cloud is currently not possible as not necessary for our use cases.

The data to be handled consist of actual time series lists (e.g., periodical measurements of smart meters or any other sensors) and meta data of the sources data point including a description, location, other meta data and statistical values based on the time series data, e.g., for detection of abnormal values. Meta data can further be populated by adding tags to describe a data point and are persisted to the local storage periodically. After insertion of a measurement, the data record stays immutable; while meta data tags may be altered at any time.

Statistics Statistical information i.e., for measurement values, frequency of updates and deltas between a measurement's time stamp and the receiving time at the storage module are

available. However, they are limited to information which can be calculated without having all (previous) values available (on-line algorithm). For each data point, the number of values, the minimum and maximum value, the mean value, variance and standard deviation are available and updated on each received entry. A histogram classifies received values in bins, e.g., of size 2.5 V for voltage measurement values. Additionally, a 95 and a 99 % confidence interval for the mean value is available. These statistics are available for each data point.

3.2.4 Storage module

The storage module is a Gridlink module responsible for storing measurement values, grid topology data and meta data. It is a vital module in an intelligent substation system as other application modules require access to historical measurement such as voltages or switch positions and other data (cf. R8). Requests issued to this module are either ones that add additional data to the storage or to request persisted data for the use of the issuing module. The storage module is primarily implemented by using Storacle for time series measurement values and the meta data. Data of which Storacle is not capable of persisting such as topology files, are either handled by the Storage module itself or other data base solutions may be used for those.

All request types the storage module is capable to handle are listed next. Most of these request types have an corresponding reply type. Some requests, naturally, reply only by returning a *SuccessReply* or *ErrorReply* (e.g., on *UpdateTagsRequest*) or are defined to not reply at all (e.g., on *AddEntryRequest*).

- *CreateDataPointRequest* to create the data point,
- *ExistsDataPointRequest* to detect whether the data point (already) exists,
- *GetDataPointsRequest* to retrieve all data points. The request may optionally contain a list of tags, such that the storage returns only those data points that contain any or all of the tags.
- *GetTagsRequest* to retrieve the tags of the data point,
- *UpdateTagsRequest* to update the tags of the data point,
- *AddEntryRequest* to add an entry of the data point to the storage,
- *GetMostRecentEntryRequest* to retrieve the most recent entry of the data point,
- *GetEntriesOfTimeSpanRequest* to retrieve all entries within a time span of the data point,
- *GetStatisticsRequest* to retrieve the statistics of the data point,
- *GetTopologyRequest* to retrieve the grid topology,
- *SaveDataPointsCSVRequest* to export the values of data points within a time span to a CSV file.

Observing storage's events Events of data points are published to designated topics, regardless whether or not anyone has interest in these events. An explicit (and traditional) registration to data point's events is therefore not necessary. To get informed of such events, it is only necessary to register to the designated data point's observer topic. The storage module issues events whenever a data point was created, a measurement was added or tags have been altered. This functionality can be used to implement a push mechanism to update a view once a new value is received. Some types of events are published to the global observer topic allowing to observe events of general interest without prior knowing a data point's name.

3.2.5 Grid representation module

The grid representation module (GRM) collects grid-data, preprocesses this data and provides current information about the distribution grid and its elements to other modules in a well structured form. The grid topology is provided as XML-based *common information model (CIM)* [20] and imported by the GRM building up a topological model using the information about nodes, lines, transformers, connections, etc. For each device, an entry in a table is created including device specific information and a unique identifier.

Typically, an adjacency matrix is used to store the information if two elements within a network are connected. In the Grid Representation Module, the matrix can be interpreted as extended adjacency matrix because it contains much more information about the topology and the relation between network elements (e.g., if they are directly connected, indirectly connected, the path length in terms of hops between two elements, and if it is possible to reach an element when changing switch positions). By using the row and column index as unique identifier of the elements, the connections information can be read very fast. The matrix is build initially when GRM is started and the topology information is available. Therefore, well-defined starting points of the grid are identified (e.g., MV/LV connection point). Starting at these elements, a recursive algorithm iterates through the elements of the network to find some predefined final nodes (e.g., energy consumers). Based on the traversed path within the network topology, the connection information is stored. By using positive and negative connection lengths, parent-child-relations in terms of network topology are shown. In case of topology changes (e.g., change of a switch position), the information is updated in the Grid Representation Module. Thus, the GRM can be seen as an up-to-date representation of the grid. Detailed information about the connection information and example can be found in [9].

Another function of the GRM is to model grid monitoring devices, in particular their position within the topology and to provide their corresponding power profiles. Therefore, the monitoring devices are identified and their power profiles are

requested from the storage module by the GRM. In general, grid monitoring devices are sensing active and reactive power flow for each phase of a line. Thus, measured power is the sum of all sub-branches and all directly connected nodes. GRM analyzes the determined positions of the monitoring devices and—in case that a parent-child-relation exists to a sub-branch—so called residual profiles are calculated. To provide useful data, gaps within the profiles are identified and seamless time-series are created.

3.2.6 Building representation module

The functionality of the Building Representation Module (BRM) is similar to the GRM—collect and preprocess data and provide it to other modules and applications. In almost the same manner as the GRM, the BRM requests profiles from the storage module, in particular profiles of buildings instead of monitoring devices within the distribution grid. Also, gaps within the profiles are identified and seamless time-series are created. As a result, suitable data can be provided for other modules of the application frame. At the moment, the building representation by their power profiles and data preparation are the only implemented functions of BRM. Further, functions like in-building actuator modeling and energy market participation-related functions are envisioned.

4 Demonstration

In this section we show the full process flow of an smart grid application deployment and operation.

4.1 Demonstration of provisioning

This section shows the installation and configuration of the involved Gridlink components. Assumed is an already running iSSN installation, some modules handling other use cases already in service. As a central module required for nearly all use cases, it can be assumed that the described *Storage* module have already been installed and is running. At some point, the iSSN administrator decides to install a new module on the iSSN, namely the *BEMS-AA* module which functionalities will be described in the next subsection. Figure 5 depicts the necessary steps to provision the *BEMS-AA* module.

1. The installation of *BEMS-AA* is initiated by the iSSN administrator using the utility dashboard.
2. Dependencies on *BRM*, *GRM* and *Storage* module are detected by the dashboard's backend.
3. The necessity to additionally install *BRM* and *GRM* is reported to the operator on the dashboard. A configura-

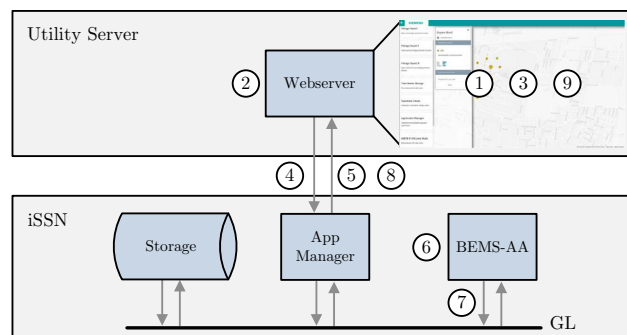


Fig. 5 Provisioning of BEMS-AA. To simplify the figure, installation of the dependencies is omitted

tion for all three apps (*BEMS-AA*, *BRM* and *GRM*) can be supplied. To simplify further explanation, note that in the following steps we refer only to “the modules”.

4. The *AppManager* is informed via XMPP to install the modules by sending the download location of the artifacts and the supplied configurations.
5. Via HTTP the *AppManager* downloads the artifacts from the backend.
6. After downloading the artifacts, they are extracted and the supplied configurations are used to start the modules according to the dependency graph.
7. During start-up, the modules register themselves to the system by joining the Hazelcast cluster and including their information in the Gridlink registry.
8. The occurrence of a module in the registry is assumed to imply its successful installation. Once the *AppManager* detects a registration by observing the registry it reports the successful installation to the dashboard backend via XMPP. In error case, the *AppManager* reports the failure of the installation after waiting some time to appear. As it holds a process list of modules started by the *AppManager* it could kill the process if necessary.
9. At the dashboard the successful installation of *BEMS-AA* is shown to the operator.

Module configuration—including the use of Gridlink proxies—or upgrades are among subsequent steps achievable by the described utility dashboard and communicated to the *AppManager* using XMPP.

4.2 Demonstration of exemplary application

This sections shows an exemplary application, featuring the implementation of the presented concept. It is based on some basic applications (cf. [9]) namely a *Storage* module, *GRM*, and *BRM* providing information about the grid topology and (rehashed) measurement values to other modules. In the SCDA—*Smart City Demo Aspern* [1] project context the so

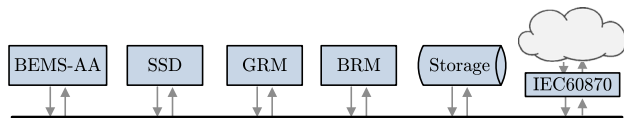


Fig. 6 SCDA application frame (cf. [9])

called *SCDA Application Frame* has been developed, consisting of the basic applications, extended by two new modules:

1. The *building energy management system assignment application (BEMS-AA)* is used to assign measured power profiles of buildings to profiles from the distribution grid. In particular, this algorithm can be used to identify the position of buildings within the grid. Therefore, a set of measured power profiles from buildings and a set of profiles from measurement points within the grid are transferred to the assignment algorithm. Based on several configuration parameters and comparison methods the best assignment is determined.
2. To verify the distribution grid topology an application called *switch state detection (SSD)* was developed which checks the correctness of current switch state information based on power profiles and the distribution grid topology.

Figure 6 depicts an overview of the SCDA Application Frame, consisting of the Storage module, Building and Grid Representation Modules, Switch State Detection, and Building Energy Management Systems Assignment Application using Gridlink to exchange data between these modules. Measurement data from the field is transferred to the SCDA Application Frame via an IEC 60870 gateway. Figure 7 shows the sequence diagram of the SCDA Application Frame to illustrate the behavior of the systems and the data transfer between the modules, consisting of the following parts:

- A. BEMS-AA is started and the necessary initialization steps are executed. Next, a data request is sent to GRM (1.a) and BRM (1.b).

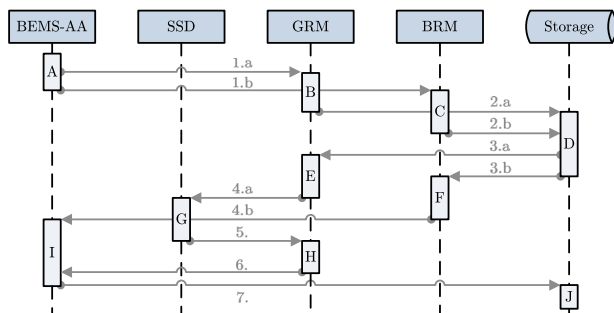


Fig. 7 SCDA application frame—sequence diagram

- B. GRM receives the data request from BEMS-AA and sends a request to the Storage module (2.a) to claim power profiles from the distribution grid.
- C. BRM receives the data request from BEMS-AA and sends a request to the Storage module (2.b) to claim power profiles from the buildings.
- D. The Storage module receives the requests from GRM and BRM and replies with grid profiles (3.a) and building profiles (3.b).
- E. GRM receives the grid profiles from the Storage module and preprocesses the data (e.g., gap-identification and synchronization). To verify the grid topology the power profiles are forwarded to SSD (4.a).
- F. BRM receives the building profiles from the Storage module, preprocesses the data (e.g., gap-identification and synchronization), and forwards them to the assignment application (4.b).
- G. SSD receives grid data from GRM and verifies the given topology by comparing the profiles within each branch of the distribution grid. The result of that verification is appended to the grid data and send back to GRM (5.).
- H. GRM receives the grid measurement data including the verification result from SSD. If the grid topology is valid, then the power profiles are forwarded to the BEMS-AA (6.), otherwise an empty set is send.
- I. BEMS-AA receives profiles from BRM and GRM. If the grid data is valid, the assignment algorithm is executed and the result is sent to the Storage module (7.). On the other hand, invalid grid data cannot be used for any assignment and thus, no data will be sent to the Storage module.
- J. Results are persisted at the Storage module.

Obviously, there is a high interaction and data transfer between the Gridlink modules. Thus, the requirement for a high performance communication infrastructure arises.

5 Conclusion and outlook

This work proposes a versatile framework for Smart Grid applications that are processed on the level of a smart secondary substations. Substation automation on the secondary level is a prime technological challenge associated with Smart Grids, as the number of these substations is too large for a single operator to oversee and manually control. Driven by renewable energies and the roll out of Smart Metering infrastructure, a strong need for cost-effective application frameworks for secondary substations will be evident in the upcoming years. We have identified major requirements from experience in practical work with distribution system operators, resulting in a modular design supported by a domain-specific data model that supports distributed appli-

cations. Further, we have shown the implementation of a fully functional software ecosystem that is capable of deployment, provisioning, automatic configuration, and effective operation of future smart grid applications. Its backbone is the distributed Gridlink middleware, that connects an efficient data storage system and arbitrary applications. The system has proven its suitability in its first applications and will be used for further application deployment and operation in the smart city context in the related research project [1] and beyond.

Acknowledgements This work received funding within the *Smart Cities* program of the Austrian *Climate and Energy Fund* in the research project *SCDA—Smart City Demo Aspern* under Project Number 846141.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Aspern Smart City Research GmbH & Co KG (ASCR) SCD—Smart City Demo Aspern (2016). <http://www.ascr.at/foerderungen/scda/>
2. Cejka S et al (2015) Java embedded storage for time series and meta data in Smart Grids. In: 2015 IEEE international conference on smart grid communications, pp 434–439 (2015)
3. CEN-CENELEC-ETSI (2012) Smart grid reference architecture. Technical report, CEN-CENELEC-ETSI smart grid coordination group. http://ec.europa.eu/energy/sites/ener/files/documents/xpert_g
4. Claassen A et al (2011) Application of the OPC UA for the smart grid. In: 2011 IEEE PES international conference and exhibition on innovative smart grid technologies, pp 1–8 (2011)
5. Colombo AW et al (2014) Industrial cloud-based cyber-physical systems—the IMC-AESOP approach. Springer, Berlin (2014)
6. Farhangi H (2010) The path of the smart grid. *IEEE Power Energy Mag.* 8(1):18–28
7. Faschang M et al (2013) Rapid control prototyping platform for networked smart grid systems. In: 39th annual conference of the IEEE Industrial Electronics Society, pp 8172–8176 (2013)
8. Faschang M et al (2016) Functional view of a smart city architecture: the SCDA greenfield approach. In: 2016 workshop on modeling and simulation of cyber-physical energy systems
9. Faschang M et al (2016) iSSN application frame—a flexible and performant framework hosting smart grid applications. In: CIRED workshop 2016. Paper 255
10. Gungor VC et al (2013) A survey on smart grid potential applications and communication requirements. *IEEE Trans Ind Inf* 9(1):28–42
11. Hazelcast FAQ. <http://docs.hazelcast.org/docs/3.5/manual/html/faq.html>. Accessed 13 April 2016. (Online)
12. Huang AQ et al (2011) The future renewable electric energy delivery and management (FREEDM) system: the energy internet. *Proc IEEE* 99(1):133–148
13. Liserre M et al (2010) Future energy systems: integrating renewable energy sources into the smart power grid through industrial electronics. *IEEE Ind Electron Mag* 4(1):18–37
14. Martini L (2015) Trends of smart grids development as fostered by European research coordination: the contribution by the EERA JP on smart grids and the ELECTRA IRP. In: 2015 IEEE 5th international conference on power engineering, energy and electrical drives, pp 23–30 (2015)
15. Pike R et al (2005) Interpreting the data: parallel analysis with Sawzall. *Sci Program* 13(4):277–298
16. Rifkin J (2013) The third industrial revolution: how lateral power is transforming energy, the economy, and the world. Macmillan, New York
17. Stifter M et al (2015) Smart meter test stand for requirement analysis of advanced smart meter applications. In: 2015 international symposium on smart electric distribution systems and technologies, pp 547–552
18. Strasser T et al (2013) Review of trends and challenges in smart grids: an automation point of view. In: Mařík V, Lastra JLM, Skobelev P (eds) Industrial applications of holonic and multi-agent systems, no. 8062. In: Lecture notes in computer science. Springer, Berlin, pp 1–12 (2013)
19. Strasser T et al (2015) A review of architectures and concepts for intelligence in future electric energy systems. *IEEE Trans Ind Electron* 62(4):2424–2438
20. Usler M et al (2012) The common information model CIM: IEC 61968/61970 and 62325—a practical introduction to the CIM. Springer Science & Business Media, Berlin (2012)
21. Vrba P et al (2014) A review of agent and service-oriented concepts applied to intelligent energy systems. *IEEE Trans Ind Informat* 10(3):1890–1903
22. Yan Y et al (2013) A survey on smart grid communication infrastructures: motivations, requirements and challenges. *IEEE Commun Surv Tutor* 15(1):5–20
23. Yu X et al (2016) Smart grids: a cyber-physical systems perspective. *Proc IEEE* 104(5):1058–1070



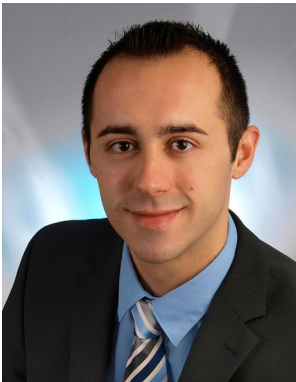
Mario Faschang is research engineer and project manager at the Energy Department of AIT Austrian Institute of Technology GmbH in Vienna. Faschang holds a M.Sc. (Dipl.-Ing.) degree (2011) in electrical engineering and information technology from TU Vienna and a Ph.D. in engineering science (2015). During his studies, Faschang worked on coordinated charging of battery-electric vehicles and rapid prototyping of distributed control systems for networked smart grid

systems. He was with Siemens AG Austria and also employed as research assistant at TU Vienna. Faschang's current research focus on future power grids and voltage control in low voltage distribution grids with high share of distributed, alternative power generation and electro mobility. Faschang was awarded the Austrian INiTS award, the "green tech" award, and the Austrian Smart Grids Pioneer Award together with his colleagues in 2012. He is member of Austrian Electrotechnical Association (OVE), IEEE Austria, and officer of IEEE Austria Young Professionals Affinity Group.



Stephan Cejka BSc joined the department for Corporate Technology of Siemens Austria in 2014. There he works as Junior Expert in a research group for Smart Grids and Smart Buildings. In 2013, he received his bachelors degree in Software & Information Engineering from Vienna University of Technology. He is currently finishing his master studies in Software Engineering & Internet Computing at Vienna University of Technology in the distributed systems area.

In 2016, he received his magister degree in Laws from University of Vienna, where he recently started his PhD studies. His PhD thesis in Laws will be located in the privacy problem area of Smart Grids. Research interests include distributed systems, data storage and privacy in the Smart Grid area.



Mark Stefan studied Computer Science (Bachelor and Master) at the Vienna University of Technology. He started his professional career at Robert Bosch AG in Vienna (software and function development, project management) where he was working for about 2.5 years. In 2012, he joined the Institute of Computer Aided Automation at the Vienna University of Technology, working as project assistant and doing his PhD-studies. He developed an algorithm for optimizing rail-

way systems in terms of deadlock detection and avoidance as well as the minimization of the traction energy consumption. Since June 2014, he is working as Research Engineer and Project Manager at the AIT Austrian Institute of Technology GmbH. Since 2014, Dr. Stefan holds lectures at the Fachhochschule St.Pölten (Application of Graphs in the Railway Sector).



Albin Frischenschlager was born in 1988 in Vienna. He received a master degree in Computer Engineering from the Vienna University of Technology. His diploma thesis was about: "Autonomous path planning using probabilistic maps". During his studies, Albin Frischenschlager was employed as system-administrator and software developer for different companies. Since October 2014, Albin Frischenschlager is working in a Smart Grid research

group of Corporate Technology at Siemens AG Austria as project manager and software developer.



Alfred Einfeld was studying electrical engineering with focus on electrical power systems at University of Technology in Vienna Austria. Afterwards he was working at the Institute of Power Systems and Energy Economics within several Smart Grid research projects as an expert and project manager. His PhD thesis was about: "Reliability of the energy supply in sustainable, autonomous Micro Grids". Since June 2011 he is employed within a research

group of Corporate Technology at Siemens AG Austria. In his role as Senior Key Expert for Smart Grid technology he is working as project manager and power systems expert in several R&D projects.



Konrad Diwold received a master degree in Artificial Intelligence from the Free University Amsterdam (Netherlands) in 2007. From 2008 until 2011 Konrad Diwold was a research associate at the parallel and complex systems workgroup at the University of Leipzig (Germany). His work concerned biological inspired algorithms. He received a Ph.D. in computer science from the University of Leipzig in 2012. From 2011 until 2014 Konrad Diwold was a research

associate at the Fraunhofer Institute for Wind Energy and Energy System Technology (Kassel, Germany) in the department of distribution system operation. His work concerned the smart integration of renewable energy resources in distribution system operation. Since January 2015, he is working in a research group of Corporate Technology with focus on Smart Grid technologies at Siemens AG Österreich.



Filip Prösl Andrén studied Applied Physics and Electrical Engineering at Linköping University from 2004 to 2009 with a thematic focus on Control and Information Systems where he received a master's degree. Since 2009 he is working as a scientist at the AIT Austrian Institute of Technology, Energy Department. He is specialized on Smart Grids and power utility automation. His main research interests are automation and control systems, communication and

automation standards as well as modeling, simulation and development of intelligent grid components.



Thomas Strasser received the Ph.D. degree in mechanical engineering, with a focus on automation and control theory, from Vienna University of Technology, Vienna, Austria, in 2003. He was a Senior Researcher with PROFACTOR Research, Steyr, Austria, working in the field of reconfigurable automation for more than six years. Since 2010, he is a Senior Scientist with the AIT Austrian Institute of Technology, Vienna, Austria, working in the domain of smart grids, with

a focus on power utility automation. He is involved in several standardization activities like IEE SC65B/WG15, IEC TC65/WG17, IEC SyC Smart Energy/WG6, and IEEE P2660.1. In addition, Dr. Strasser is a senior member of the IEEE and Associate Editor of IEEE and Springer journals as well as involved in several program committees of well-known international conferences (IEEE IECON, IEEE SMC, IEEE ISIE, etc.). He is also active as lecturer at local universities and international summer schools.



Friedrich Kupzog achieved the Diploma Engineer degree of electrical engineering and information technology from RWTH Aachen in 2006. After that, he joined the Institute of Computer Technology at Vienna Technical University, Austria, where he achieved his PhD Degree in 20008. Until 2012, he stayed at the University as Post-Doc and managed the research group “Energy & IT” at the Institute of Computer Technology. Since 2012, Dr. Kupzog is Senior Scientist at the AIT Austrian Institute of Technology GmbH. His research

interest lies in verification methods for networked smart grid systems. He coordinates the thematic field “Smart Grids ICT & Controls”, managing research projects together with industry, power grid operators and other research partners. Dr. Kupzog holds lectures in smart grid related topics at Vienna University of Technology as well as Technikum Vienna and is active in national (ComForEn) and international (IEEE IECON, IEEE/CIGRE EDST, D-A-CH Energieinformatik) scientific conference organisation. He was awarded the Austrian Smart Grid Pioneer Award together with his colleagues in 2010 and 2012.